Discrete Applied Mathematics 13 (1986) 9–22
North-Holland

AN ASSIGNMENT ALGORITHM WITH APPLICATIONS TO INTEGRATED CIRCUIT LAYOUT*

Mikhail J. ATALLAH and Susanne E. HAMBRUSCH
Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA

Received 12 March 1985
Revised 27 August 1985

The 2-terminal one-to-any problem, which arises in the design of layout systems, is the problem of assigning each one of n terminals positioned on the upper row of a channel (called *entry terminals*) to one of m terminals positioned on the lower row (called *exit terminals*) so that the resulting channel routing problem has minimum density. An optimal solution to this problem is known [1]. In this paper we consider a natural generalization, the 2-color one-to-any problem, in which we have two types of entry terminals, red and blue ones, and exit terminals can be assigned to either type of entry terminal. Red and blue nets created by our algorithm are allowed to run on top of each other in the routing, and the density is defined as the larger of the red density and the blue density. Its minimization is an interesting combinatorial problem. We show how to compute the best achievable density in $O(n+m)$ time, and an assignment achieving this density in $O((n+m)\log(n+m))$ time.

Keywords. Analysis of algorithms, channel routing, data structures, density, terminal assignments.

1. Introduction

The 2-terminal one-to-any terminal assignment problem is that of assigning each one of n entry terminals positioned on the upper row of a channel to one of m exit terminals positioned on the lower row so that the resulting channel routing problem (CRP) consisting of n nets has minimum density. This problem arises in the design of layout systems for integrated circuits: When it is irrelevant which exit terminal an entry terminal is connected to (or when the modules associated with the exit terminals can be arranged in any order), this freedom should be used to generate a CRP that requires minimum width between the two rows in the subsequent routing stage [5,6]. For a number of routing models the channel width required by a routing algorithm is proportional to the density of the CRP [4,7,9]. In [1] we presented an algorithm that computes, in $O(n+m)$ time, the best achievable density for a 2-terminal one-to-any problem, as well as an assignment achieving this density.

This paper considers a natural generalization to the above problem, namely the 2-color one-to-any problem, in which we have two types of entry terminals, red and

* This work was supported by the Office of Naval Research under Contract N00014-84-K-0502, and by the National Science Foundation under Grants DCR-84-51393 and DMC-84-13496.

blue ones, and exit terminals can be assigned to either type of entry terminals. The assignment of an exit terminal to a red (resp. blue) entry terminal creates a red (resp. blue) net. This generalization has an interesting combinatorial structure, and it is motivated by wiring models that consist of multiple layers which allow red and blue nets to be routed independently on different layers. Thus, the density to be minimized is defined differently from the 1-color problem. We show how to compute the optimal density for a 2-color problem in $O(n+m)$ time, and how to obtain an assignment achieving the optimal density in $O((n+m)\log(n+m))$ time.

Before giving a more formal definition of the 2-color problem, we define the (1-color) one-to-any problem. The positions of n entry terminals on the upper row of the channel are $p_1, \ldots, p_n$, $1 \le p_1 < \cdots < p_n \le M$, and the positions of m exit terminals on the lower row of the channel are $q_1, \ldots, q_m$, $1 \le q_1, \ldots, q_m \le M$, $m \ge n$. We have to compute a one-to-one fuction $f, f: [1:n] \to [1:m]$, so that the CRP consisting of the pairs, called *nets,* $(p_1, q_{f(1)}), \ldots, (p_n, q_{f(n)})$ has minimum density. The *density* d of a CRP is the maximum over all x of the number of nets $(p_i, q_{f(i)})$ for which $p_i \le x < q_{f(i)}$ or $q_{f(i)} > x \ge p_i$. An equivalent formulation of the 1-color one-to-any problem is the following: The p_i's and q_i's correspond to positions on a horizontal line, and the problem is to create n intervals whose endpoints are p_i and $q_{f(i)}$, $1 \le i \le n$, so that the amount of overlap between the created intervals is minimized. (The overlap is the largest number of intervals cut by any vertical line, and it corresponds to the density in our terminology.)

The *dynamic* (1-color) one-to-any problem is that of maintaining, on-line, the optimal density when exit and entry terminals are added and deleted. In Section 2 of this paper we sketch a data structure which solves this problem in $O(\log(m+n))$ time per insertion/deletion, and uses $O(n+m)$ space. This data structure will be a useful tool in Sections 3 and 4, which present the solution to the 2-color one-to-any problem. A precise definition of the 2-color one-to-any problem follows.

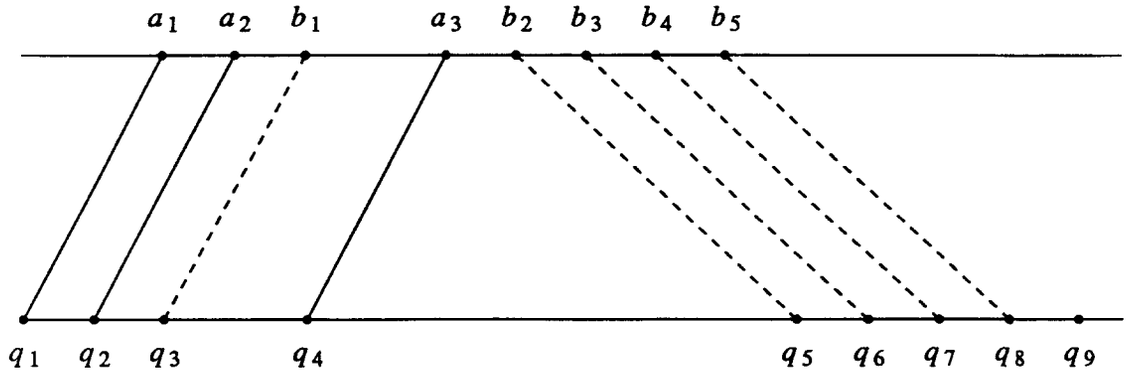
In the *2-color one-to-any problem* every entry terminal is either of type red or type blue, while exit terminals are all of the same type and can be assigned to either a red entry terminal or to a blue one. The routing algorithm, which follows our assignment algorithm, allows nets of different type to be routed independently (i.e., they can run on top of each other). Therefore, the density to be minimized by our algorithm is defined as the larger of the density for the red nets and that for the blue nets. More formally, let the positions of the red entry terminals be $a_1, \ldots, a_r$, and the positions of the blue entry terminals be $b_1, \ldots, b_s$, $r+s=n$, and let the m exit terminals be $q_1, \ldots, q_m$, $m \ge n$. We now have to compute two one-to-one functions f and g, $f: [1:r] \to [1:m]$ and $g: [1:s] \to [1:m]$ with $f(i) \ne g(j)$ for every i and j, so that the 2-color CRP consisting of the red nets $(a_1, q_{f(1)}), \ldots, (a_r, q_{f(r)})$ and the blue nets $(b_1, q_{g(1)}), \ldots, (b_s, q_{g(s)})$ has minimum red/blue density. The *red/blue density* of a CRP is the larger of d_a and d_b, where d_a is the density of the red nets and d_b is the density of the blue nets. Just as for the 1-color problem, this problem can also be phrased in terms of intervals on a line: It can be viewed as the problem of creating

r red intervals $\{a_1, q_{f(1)}\} \cdots \{a_r, q_{f(r)}\}$ and s blue intervals $\{b_1, q_{g(1)}\} \cdots \{b_s, q_{g(s)}\}$ so as to minimize the larger of the red overlap and the blue overlap. (Note that $\{x, y\}$ does not imply $x \leq y$.)

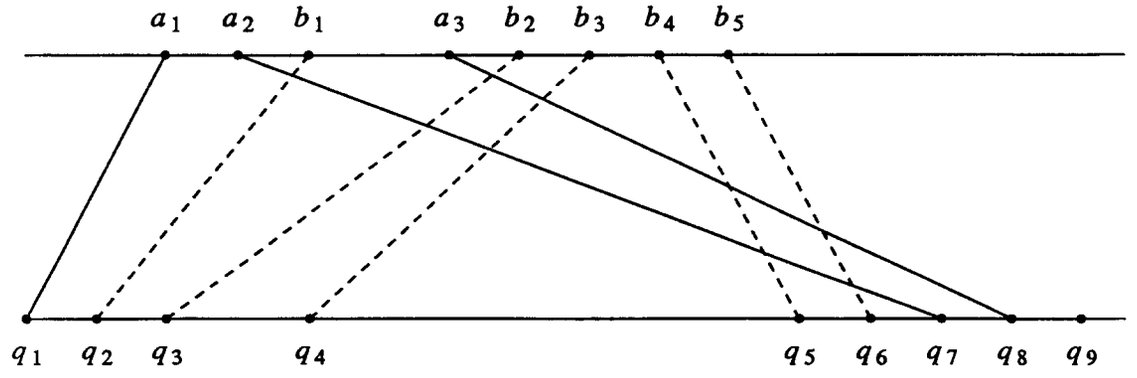
Fig. 1(a) shows an assignment having a red/blue density of 4, while Fig. 1(b) shows an optimal assignment having a red/blue density of 2. Observe that, in Fig. 1(a), assigning q_1 to a_1 and q_2 to a_2 makes it impossible to achieve red/blue density 2, while a red/blue density of 3 is still possible. In the 2-color problem red and blue terminals compete for exit terminals, and the strategy will be to assign an exit terminal to the color that 'needs it more'. We show how to compute the optimal red/blue density $\hat{d}$ in $O(n+m)$ time, and how to generate in $O((n+m)\log(n+m))$ time an assignment of entry to exit terminals that achieves $\hat{d}$.

2. Preliminaries

In this section we introduce some terminology and we describe a data structure for solving the dynamic 1-color one-to-any problem. Recall that in this problem we maintain the value of the smallest achievable density of a one-to-any problem when



(a) assignment with d_a=1 and d_b=4



(b) assignment with d_a=2 and d_b=2

Fig. 1.

entry and exit terminals are added and deleted. Our implementation uses a balanced tree structure and supports each operation in $O(\log(n+m))$ time. We also make use of a result described by the authors in reference [1] which is briefly summarized next.

Definition 2.1. For all x and y, $1 \le x \le y \le M$, we define $up(x, y)$ to be the number of entry terminals whose position is $\ge x$ and $\le y$. We define $down(x, y)$ to be the number of exit terminals whose position is $\ge x$ and $\le y$. Let $out(x, y) = up(x, y) - down(x, y)$.

Intuitively, if $out(x, y)$ is positive, it is equal to the number of entry terminals between and including x and y that have to be assigned to exit terminals at positions $< x$ or $> y$. In [1] we have shown that $\hat{d}$, the optimal density for a 1-color one-to-any problem, can be expressed as in Lemma 2.2 and computed in $O(n+m)$ time.

Lemma 2.2. $\hat{d} = \max\{\max_{x,y} \lceil out(x, y)/2 \rceil, \max_x out(1, x), \max_x out(x, M)\}$.

Our solution to the dynamic one-to-any problem uses a balanced binary tree T [2] whose leaves contain the positions of entry and exit terminals merged together in sorted order. The entries of each internal node v in T contain, in addition to the usual balanced tree entries, information about the density of the one-to-any problem consisting of the entry and exit terminals currently stored in the subtree rooted at v. To store this information each node v contains four integer entries $l(v)$, $m(v)$, $r(v)$, and $c(v)$. Let x_0(resp. y_0) be the position of the leftmost (resp. rightmost) terminal in the subtree of v. Note that x_0 (resp. y_0) may coincide with an entry terminal, with an exit terminal, or with both. Then,

$$l(v) = \max_{x_0 \le x \le y_0} out(x_0, x),$$
$$r(v) = \max_{x_0 \le x \le y_0} out(x, y_0),$$
$$m(v) = \max_{x_0 \le x \le y \le y_0} out(x, y),$$
$$c(v) = out(x_0, y_0).$$

If v is the root, then it follows from Lemma 2.2 that $\hat{d} = \max\{\lceil m(v)/2 \rceil, l(v), r(v)\}$.

The above definitions of $l(v)$, $r(v)$, $m(v)$, and $c(v)$ imply that, if w_1 and w_2 are the left and right child of v, respectively, then:

$$l(v) = \max(l(w_1), c(w_1) + l(w_2)),$$
$$r(v) = \max(r(w_2), r(w_1) + c(w_2)),$$
$$m(v) = \max(m(w_1), m(w_2), r(w_1) + l(w_2)),$$
$$c(v) = c(w_1) + c(w_2).$$

The correctness of the above relationships follows from the definition of the four entries. The relationships allow us to compute, in time $O(1)$, the information about

the density at a node from that at its children. This means that the effect of an addition or deletion of a terminal on the optimal density can be propagated from the appropriate leaf to the root in time proportional to the height of the tree (further implementation details are left to the reader). It follows that each operation can be performed in $O(\log(n+m))$ time, where n is the number of entry terminals and m the number of exit terminals currently in the tree.

3. A preliminary algorithm for the 2-color problem

In this section we present an $O((n+m)\log(n+m))$ time algorithm, called the *verification algorithm*, which determines, for any given integer d, whether a 2-color one-to-any problem has a solution of red/blue density $\leq d$. The verification algorithm immediately leads to an $O((n+m)\log(n+m)\log \hat{d})$ time algorithm for computing the optimal density $\hat{d}$ and an optimal assignment (i.e., one achieving $\hat{d}$) by $\log \hat{d}$ applications of the verification algorithm. In the next section we show how to compute $\hat{d}$ in $O(n+m)$ time and thus eliminate the $\log(n+m)\log \hat{d}$ factor in the time complexity for computing $\hat{d}$. Without loss of generality we can consider assignments in which no two red (resp. blue) nets cross each other; i.e., if $i<j$ then $f(i)<f(j)$ (resp. $g(i)<g(j)$). This is easily seen by noting that whenever two red (resp. blue) nets (x_i, q_α) and (x_j, q_β) with $x_i<x_j$ cross each other, replacing these two nets by (x_i, q_β) and (x_j, q_α) cannot increase the red (resp. blue) density. We first define some additional terminology.

By a (partial) assignment $P_{i,j,l}$, $i+j\leq l$, we mean one in which every one of the $i+j$ entry terminals $a_1, \ldots, a_i$ and $b_1, \ldots, b_j$ has been assigned an exit terminal, and q_l is the rightmost assigned exit terminal. In other words, q_l is assigned to either a_i or b_j, and all a_k's that are $>a_i$ and b_k's that are $>b_j$ have no exit terminal assigned to them yet. By convention, $P_{0,0,0}$ denotes the empty assignment. We omit the subscripts in $P_{i,j,l}$ when their value is irrelevant or is clear from the context.

Definition 3.1. If $i=r$, then assignment $P_{i,j,l}$ is *red-complete*, if $j=s$, then it is *blue-complete*, and if both $i=r$ and $j=s$, then we simply say that it is *complete*. An assignment $P_{i,j,l}$ is a *d-assignment* if the red/blue density resulting from the $i+j$ nets created so far is $\leq d$.

Definition 3.2. A d-assignment $P_{i,j,l}$ is *left-compressed* if the following property holds for every net (v, w) in $P_{i,j,l}$: If q is an unassigned exit terminal to the left of w, then replacing (v, w) by (v, q) would cause the density to exceed d.

It should be clear that if there exists a d-assignment, then there exists one which is left-compressed. From now on all assignments, whether they are partial or complete, are assumed to be left-compressed.

Definition 3.3. An assignment $\hat{P}$ is an *extension* of $P_{i,j,l}$ if $\hat{P}$ is identical to $P_{i,j,l}$ after all the nets that use an exit terminal at a position $>q_l$ have been removed from $\hat{P}$.

If P is an assignment in which entry terminal v and exit terminal w are both unassigned, then $P+(v, w)$ denotes the assignment obtained by adding the net (v, w) to P. Similarly, if (v, w) is a net in assignment P, then $P-(v, w)$ denotes the result of removing (v, w) from P.

Informally, the verification algorithm works by scanning simultaneously each of the three input lists: the list of the red entry terminals, the list of the blue entry terminals, and the list of the exit terminals. At each exit terminal we decide whether to assign it to the red entry terminal currently scanned, to the blue entry terminal currently scanned, or to leave it unassigned. In order to make this decision we compute a red and a blue label whose meanings are discussed below.

The *red label*, called α, and the *blue label*, called β, of the exit terminal q_{k+1} with respect to the d-assignment $P_{i,j,l}$, $l \leq k$, are defined when the following two conditions hold:

C1: $P_{i,j,l}$ can be extended into a *red*-complete assignment $P1$ whose red density is $\leq d$ (recall that in $P1$ we are allowed to use the exit terminals $q_{f(1)}, q_{f(2)}, \cdots, q_{f(i)}, q_{k+1}, q_{k+2}, \cdots, q_m$), and

C2: $P_{i,j,l}$ can be extended into a *blue*-complete assignment $P2$ whose blue density is $\leq d$.

If C1 and C2 are true, then the red label α is one of $\{yes, no, \#\}$. It is *no* if $P_{i,j,l}+(a_{i+1}, q_{k+1})$ has density $d+1$. Observe that the fact that a red-complete assignment $P1$ exists implies $q_{k+1}<a_{i+1}$ (this in turn implies that $P1$ does not use q_{k+1}). If the red label α is not *no*, it is either *yes* or #: It is *yes* if $P1$ *has* to make use of q_{k+1} in order to achieve red density $\leq d$, and it is # otherwise. Intuitively, # can be thought of as a 'don't care' symbol, in the sense that $P1$ doesn't have to make use of q_{k+1}, but it can if it wants to. The blue label β is defined in a similar way. The *label* of q_{k+1} with respect to the d-assignment $P_{i,j,l}$ is then the pair (α, β). For example, in Fig. 1(b) $P_{1,1,2}=\{(a_1, q_1), (b_1, q_2)\}$ is a 2-assignment, and the label of q_3 with respect to $P_{1,1,2}$ is $(\#, yes)$.

Throughout, we assume that conditions C1 and C2 are satisfied at the beginning of the algorithm; i.e., the label of q_1 is defined. This initial check for C1 and C2 can be done in $O(n+m)$ time [1].

As already mentioned, the verfication algorithm creates nets while scanning the channel from left to right. The outline of the verification algorithm is given in Fig. 2. The algorithm uses a procedure *label*(q_{k+1}) which returns two values, α and β, and whose meanings have been discussed above. The implementation of procedure *label* will be discussed later in this section. Variable *NETS* used in Fig. 2 contains the set of nets created so far by the algorithm. At any time the assignment consisting

```
begin
    i,j,k := 0;
    NETS := ∅;
    (α,β) := label(q_{k+1});
    while (α,β)≠(yes,yes) and (i<r or j<s) do
    begin
        if α=yes then
        begin
        NETS := NETS+(a_{i+1},q_{k+1});
        i := i+1
        end
        else  if β=yes then
              begin
              NETS := NETS+(b_{j+1},q_{k+1});
              j := j+1
              end
              else  if α=# then
                    begin
                    NETS := NETS+(a_{i+1},q_{k+1});
                    i := i+1
                    end
                    else  if β=# then
                          begin
                          NETS := NETS+(b_{j+1},q_{k+1});
                          j := j+1
                          end;
        k := k+1;
        (α,β) := label(q_{k+1})
    end;
    if (α,β)=(yes,yes)      then write("No Complete d-Assignment Exists")
                            else output NETS
end.
```

Fig. 2. Outline of the verification algorithm.

of the nets in *NETS* has the left-compressed property and it has red/blue density $\leq d$. Suppose that the algorithm has computed a d-assignment $P_{i,j,l}$ (which is recorded in *NETS*) and that q_{k+1} is the next exit terminal to be considered, $l \leq k$. In order to decide what to do with q_{k+1} (i.e., whether to assign it to a_{i+1}, to b_{j+1}, or to leave it unassigned), the algorithm computes the label (α, β) of q_{k+1} with respect to $P_{i,j,l}$. If both α and β are *yes*, then the algorithm stops with failure (i.e., it reports that there is no complete d-assignement). If one of α or β (but not both) is *yes*, then q_{k+1} must be assigned to the color whose label is *yes*. If $\alpha = \#$ and $\beta = no$ (resp. $\alpha = no$ and $\beta = \#$), q_{k+1} is assigned to a_{i+1} (resp. b_{j+1}). If both α and β equal $\#$, the algorithm assigns q_{k+1} to a_{i+1} (we will show in Lemma 3.4 that it does not matter whether a_{i+1} or b_{j+1} is chosen). If both α and β equal *no*, then the scan 'skips' q_{k+1}. In all cases the algorithm continues with exit terminal q_{k+2} (except, of course, when it stops with failure).

Observe that conditions C1 and C2 remain both satisfied (i.e., a red- and a blue-complete assignment exists) until the moment the algorithm stops with failure. This follows from the way the algorithm works: It stops with failure because both $P1$

and $P2$ 'need' q_{k+1} in order to achieve red/blue density $\leq d$. Correctness of the above outlined algorithm is a consequence of the following lemma:

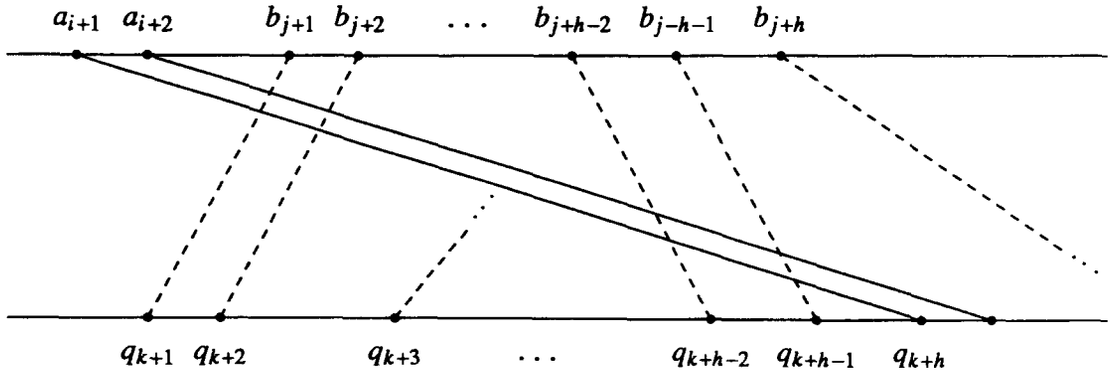
Lemma 3.4. *Let $P_{i,j,l}$ be a d-assignment and let the scan be at a_i, b_j and q_k, $l \leq k$. Suppose that this current d-assignment P can be extended into a complete d-assignment, and let (α, β) be the label of q_{k+1} with respect to P. If $(\alpha, \beta) \in \{(yes, \#), (yes, no), (\#, no), (\#, \#)\}$, then $P + (a_{i+1}, q_{k+1})$ can be extended into a complete d-assignment. Symmetrically, if $(\alpha, \beta) \in \{(\#, yes), (no, yes), (no, \#), (\#, \#)\}$, then $P + (b_{j+1}, q_{k+1})$ can be extended into a complete d-assignment.*

Proof. The only nontrivial part of the lemma is proving that, if $(\alpha, \beta) = (\#, \#)$, then it does not matter whether we assign q_{k+1} to a_{i+1} or to b_{j+1}. Either choice results in a d-assignment which can still be extended to a complete d-assignment. Without loss of generality, we assume that $a_{i+1} < b_{j+1}$.

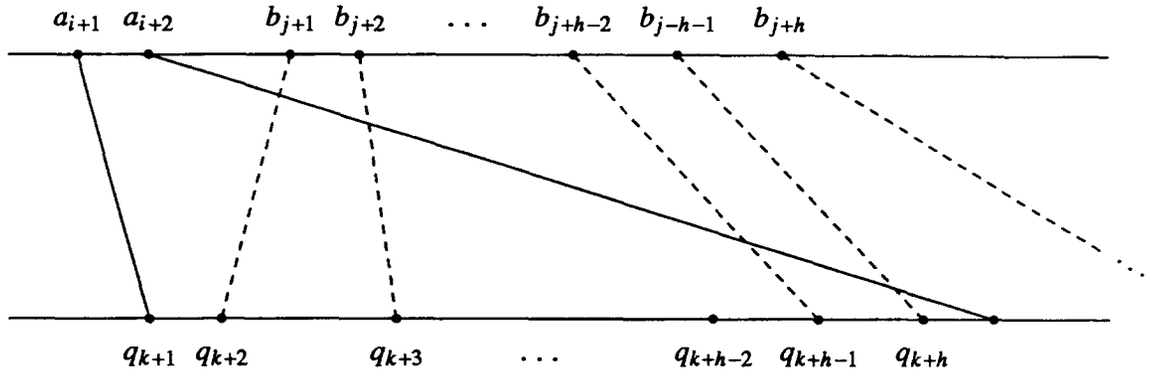
First we show that $P + (a_{i+1}, q_{k+1})$ can be extended into a complete d-assignment. Let $\hat{P}$ be a complete d-assignment that is an extension of P. Note that, since $\hat{P}$ is left-compressed and $(\alpha, \beta) = (\#, \#)$, q_{k+1} is assigned to either a_{i+1} or b_{j+1}. If, in $\hat{P}$, q_{k+1} is assigned to a_{i+1}, then we are done. Hence, suppose that q_{k+1} is assigned to b_{j+1} and that q_{k+h} is assigned to a_{i+1}, $h > 1$. Since $\hat{P}$ is left-compressed, there are no unassigned exit terminals between q_{k+1} and q_{k+h}, and $q_{k+2}, \ldots, q_{k+h-1}$ are assigned to the blue terminals $b_{j+2}, \ldots, b_{j+h-1}$, respectively. See Fig. 3(a). Suppose we replace the net (a_{i+1}, q_{k+h}) by (a_{i+1}, q_{k+1}) and remove (b_{j+1}, q_{k+1}); i.e., a_{i+1} 'grabs' q_{k+1} away from b_{j+1}. This does not cause the red density to exceed d since $\alpha = \#$. But it leaves b_{j+1} without an assigned exit terminal, and frees q_{k+h} for possible use by the blue terminals. We now show how we can 'patch up' the situation for the blue terminals without causing the blue density to exceed d: remove the nets $(b_{j+2}, q_{k+2}), \ldots, (b_{j+h-1}, q_{k+h-1})$, and create the nets $(b_{j+1}, q_{k+2}), \ldots, (b_{j+h-1}, q_{k+h})$, in that order. See Fig. 3(b). We claim that this does not cause the blue density to exceed d. Suppose to the contrary that it does, i.e. suppose that the blue density first exceeds d following the addition of the net (b_{j+u}, q_{k+u+1}) for some $1 \leq u \leq h-1$. We distinguish two cases:

Case (i): $q_{k+u+1} < b_{j+u}$. In this case d blue entry terminals at positions $> q_{k+u+1}$ and $< b_{j+u}$ are assigned to exit terminals to the left of q_{k+u+1}. In $\hat{P}$, these d blue entry terminals as well as b_{j+u} were all assigned to exit terminals to the left of q_{k+u+1}. This implies that in $\hat{P}$ the blue density at column q_{k+u+1} (i.e., between position q_{k+u+1} and $q_{k+u+1}+1$) was $d+1$, a contradiction.

Case (ii): $q_{k+u+1} > b_{j+u}$. Since assigning q_{k+u+1} to b_{j+u} causes the blue density to exceed d, there must be a column x, $b_{j+u} \leq x < q_{k+u+1}$, such that the number of blue nets that cross column x from left to right is already equal to d. It is not hard to see that $x > q_{k+1}$. Let θ be the number of blue entry terminals that are at positions $\geq b_{j+1}$ and $\leq x$, and let γ be the number of exit terminals that are at positions $> q_{k+1}$ and $\leq x$. Then we have $\theta - \gamma \geq d+1$. Thus, even if all the exit terminals to the right of q_{k+1} are colored blue, there is no hope of assigning $b_{j+1}, \ldots, b_s$ to exit



(a) before



(b) after

Fig. 3.

terminals without the blue density exceeding d. In other words, q_{k+1} is needed by the blue terminals. This implies that β, the blue label of q_{k+1} with respect to P and d, equals *yes*, a contradiction.

In a similar way, one can show that $P+(b_{j+1}, q_{k+1})$ can also be extended into a complete d-assignment. □

Now that we proved its correctness, we turn our attention to the implementation of the verification algorithm. To show that it can be implemented to run in time $O((m+n)\log(m+n))$, it clearly suffices to show that, whenever the algorithm needs a label (α, β), such a label can be computed in time $O(\log(m+n))$. We outline below how to compute label α in $O(\log(r+m))$ time. The computation of label β is similar.

Determining whether $\alpha = no$, i.e., density d is exceeded by assigning a_{i+1} to q_{k+1}, can be done in $O(1)$ time, as follows. Consider first the case when $q_{k+1} < a_{i+1}$. Let a_l be the leftmost red entry terminal with $a_l > q_{k+1}$. We set $\alpha = no$ if $i-l+1=d$. Now consider the case when $q_{k+1} > a_{i+1}$. Let a_l be the leftmost red entry terminal with $q_{f(l)} > a_{i+1}$. We set $\alpha = no$ if $i-l+1=d$. Note that the second case is mentioned here only for completeness sake: the verification algorithm would have stopped already with failure at q_{l-1} or earlier.

If $\alpha \neq no$, then we determine whether it is *yes* or # in the following way. We use a tree structure T_a as described in Section 2. The leaves of T_a contain, in left to right order, the merge of the two sets $a_1, \ldots, a_r$ and $q_{f(1)}, \ldots, q_{f(i)}, q_{k+1}, q_{k+2}, \ldots, q_m$. T_a answers the following query in $O(\log(r+m))$ time: Can $a_1, \ldots, a_r$ be assigned to the exit terminals currently in T_a *excluding* q_{k+1} without exceeding a red density of d? To implement the query we simulate the effect of deleting q_{k+1} from T_a on the red density stored in the root of T_a. If the red density increases to $d+1$, then the label of q_{k+1} is *yes*. Otherwise it is #. When the algorithm decides to assign the exit terminal q_{k+1} to a blue entry terminal, q_{k+1} is deleted from T_a, and the entries in the nodes of T_a are updated as described in Section 2. This concludes the discussion of the verification algorithm.

4. An efficient algorithm for the 2-Color Problem

As already mentioned, the $O((n+m)\log(n+m))$ time verification algorithm described in Section 3 immediately leads to an $O((n+m)\log(n+m)\log \hat{d})$ time algorithm for computing the optimal density $\hat{d}$ and a $\hat{d}$-assignment by $\log \hat{d}$ applications of the verification algorithm. The main result of this section is that the optimal density $\hat{d}$ can be computed in $O(n+m)$ time. Hence, by first computing $\hat{d}$ and then using the verification algorithm once (rather than $\log \hat{d}$ times), the assignment achieving density $\hat{d}$ can be computed in $O((n+m)\log(n+m))$ time.

We now define a quantity Φ which we will show to be equal to the optimal density of a 2-color problem. Let $up_a(x,y)$, $up_b(x,y)$, $down(x,y)$, $out_a(x,y)$, and $out_b(x,y)$ be defined as in Section 2 with respect to the red and blue entry terminals (i.e., $up_a(x,y)$ is the number of red entry terminals whose position is $\geq x$ and $\leq y$).

Definition 4.1. Let Φ be defined as follows:

$$\Phi = \max\left\{\max_x\left\{out_a(1,x), out_b(1,x), \left\lceil \frac{up_a(1,x)+up_b(1,x)-down(1,x)}{2}\right\rceil\right\}\right.$$
$$\max_{x,y}\left\{\left\lceil \frac{out_a(x,y)}{2}\right\rceil, \left\lceil \frac{out_b(x,y)}{2}\right\rceil, \left\lceil \frac{up_a(x,y)+up_b(x,y)-down(x,y)}{4}\right\rceil\right\},$$
$$\left.\max_x\left\{out_a(x,M), out_b(x,M), \left\lceil \frac{up_a(x,M)+up_b(x,M)-down(x,M)}{2}\right\rceil\right\}\right\}.$$

It is not hard to see that Φ is a lower bound on the optimal density $\hat{d}$; i.e., $\hat{d} \geq \Phi$. We only go through the argument for the third term in the second line of the definition of Φ. If $up_a(x,y)+up_b(x,y) > down(x,y)$, then at least $up_a(x,y)+up_b(x,y)-down(x,y)$ red or blue entry terminals have to be assigned to exit terminals at positions $<x$ or $>y$. The smallest density we can hope to achieve in columns $x-1$ and y, respectively, is obtained by having the number of red entry terminals assigned

to such 'outside' exit terminals equal the number of blue entry terminals assigned to 'outside' exit terminals, and by having the number of red (resp. blue) entry terminals that choose an exit terminal at a position $<x$ equal to the number of red (resp. blue) entry terminals that choose one at a position $>y$. The proof that $\hat{d}$ is no smaller than the other terms in the definition of Φ is similar and is omitted. That $\hat{d}=\Phi$ is far less obvious, and will be proved shortly.

Lemma 4.2. *Φ can be computed in* $O(n+m)$ *time.*

Proof. The value of x maximizing the first and the third line in the definition of Φ can clearly be computed in $O(n+m)$ time. That $\max_{x,y} out_a(x,y)$ can be computed in $O(n+m)$ time follows from the following observation: If $c_1,\ldots,c_N$ is a sequence of positive and negative numbers, then finding an i and a j such that $\sum_{k=i}^{j} c_k$ is a maximum can be done in time $O(N)$ (see [3] for a proof). The other two terms of line two are similarly computed in time $O(m+n)$. □

We now prove that density Φ is achievable by showing that the verification algorithm described in Section 3 always succeeds in generating a complete Φ-assignment.

Lemma 4.3. $\hat{d}=\Phi$.

Proof. We have already pointed out that $\hat{d}\geq\Phi$, so it suffices to show that a complete Φ-assignment exists. We now show that the verification algorithm of Section 3 always succeeds in generating a complete Φ-assignment. Suppose to the contrary that it stops with failure. Recall that the verification algorithm fails at a_{i+1}, b_{j+1}, and q_{k+1} having computed a Φ-assignment $P_{i,j,l}$ if the label of q_{k+1} is (*yes, yes*). We claim that this cannot happen.

That the label of q_{k+1} is (*yes, yes*) implies that there is no solution of density $\leq\Phi$ to the red (resp. blue) 1-color one-to-any problem consisting of the entry terminals $a_1,\ldots,a_r$ (resp. $b_1,\ldots,b_s$) and the exit terminals $q_{f(1)},\ldots,q_{f(i)},q_{k+2},\ldots,q_m$ (resp. $q_{g(1)},\ldots,q_{g(j)},q_{k+2},\ldots,q_m$). There are two possible reasons why the red (resp. blue) label of q_{k+1} is *yes*:

Reason A. Without q_{k+1}, there are not enough remaining exit terminals for the remaining red (resp. blue) entry terminals; i.e., $r-i=m-k$ (resp. $s-j=m-k$).

Reason B. Even though there are enough exit terminals without q_{k+1}; i.e., $r-i>m-k$ (resp. $s-j>m-k$), q_{k+1} is needed in order to extend the current assignment into a red- (resp. blue-) complete one whose red (resp. blue) density is $\leq\Phi$.

Suppose that one or both of the two (red and blue) labels of q_{k+1} is *yes* because of reason A. Consider the first instant of time during the scan at which the combined number of remaining (i.e., not yet assigned to an exit terminal) red and blue entry terminals exceeded the number of remaining exit terminals (this must have happened

at some time). Suppose that this first happened when the scan was at a_{u+1}, b_{v+1}, and q_{w+1}. We then had $r-u+s-v=m-w+1$. Let $q_{w+1-\lambda}$ be the rightmost unassigned exit terminal at a position $\leq q_{w+1}$ whose label was (*no,no*) when the scan was at $q_{w+1-\lambda}$, $\lambda \geq 0$. Note that such an exit terminal must exist (otherwise we have $r+s<m$), and that every unassigned exit terminal at a position $<q_{w+1}$ had the label (*no,no*) when it was considered during the scan. We now claim that $q_{w+1-\lambda}<a_{u+1}$ and $q_{w+1-\lambda}<b_{v+1}$. To prove the claim, simply note that if $q_{w+1-\lambda}$ is larger than $\min(a_{u+1}, b_{v+1})$, then the fact that its label is (*no,no*) means that $\min(a_{u+1}, b_{v+1})$ cannot be assigned to $q_{w+1-\lambda}$ or to any other exit terminal to its right without the density becoming $\Phi+1$. This implies that the label of $q_{w+1-\lambda}$ is not defined, a contradiction, and proves the claim.

Now, since $q_{w+1-\lambda}<a_{u+1}$ and $q_{w+1-\lambda}<b_{v+1}$, the assignment $P_{u,v,w}$ has to contain Φ red (resp. blue) entry terminals at positions $>q_{w+1-\lambda}$ that are assigned to exit terminals at position $\leq q_{w+1-\lambda}$. Let δ (resp. γ) be the number of red (resp. blue) entry terminals at positions $<a_{u+1}$ (resp. $<b_{v+1}$) that are assigned to an exit terminal at a position $>q_{w+1-\lambda}$ and $<q_{w+1}$. Then,

$$\left\lceil \frac{up_a(q_{w+1-\lambda}, M)+up_b(q_{w+1-\lambda}, M)-down(q_{w+1-\lambda}, M)}{2} \right\rceil$$

$$= \left\lceil \frac{(\Phi+\delta+r-u)+(\Phi+\gamma+s-v)-(\delta+\gamma+(m-w))}{2} \right\rceil > \Phi,$$

a contradiction. This shows that none of the two (red or blue) labels of q_{k+1} is *yes* because of reason A.

Now, assume that both labels of q_{k+1} are *yes* because of reason B. Then consider the solution S_a of density Φ to the red 1-color problem that assigns $a_1, \ldots, a_i$ to $q_{f(1)}, \ldots, q_{f(i)}$, respectively, and each of $a_{i+1}, \ldots, a_r$ to one of $q_{k+1}, \ldots, q_m$ in the left compressed fashion. Then we claim the following about S_a:

(a) S_a contains at least one column x_a of density Φ with $x_a \geq q_{k+1}$ and where the nets achieving density Φ are right nets; i.e., they have their entry terminal at a position $\leq x_a$ and their exit terminal at a position $>x_a$.

(b) Let x_a be the leftmost column in S_a satisfying property (a). Then, in S_a, all exit terminals at positions $\geq q_{k+1}$ and $\leq x_a$ are assigned to red entry terminals.

The proof of the claim is straightforward and is omitted. Let x_b be defined analogously for the blue 1-color problem. Without loss of generality we assume $x_a \leq x_b$.

Let $b_{j+1+\gamma+\Phi}$ be the rightmost blue entry terminal at a position $\leq x_b$. Since all exit terminals between q_{k+1} and x_b are assigned to blue entry terminals in S_b, the rightmost exit terminal $\leq x_b$ is $q_{k+1+\gamma}$. Let $a_{i+1+\delta_1+\Phi}$ be the rightmost red entry terminal at a position $\leq x_a$ and let $a_{i+1+\delta_1+\Phi+\delta_2}$ be the rightmost entry terminal at a position $\leq x_b$; i.e., there are δ_2 red entry terminals between x_a+1 and x_b. Assume first that all exit terminals to the left of q_{k+1} are assigned; i.e., $i+j=k$. Then,

$$\left\lceil \frac{up_a(1, x_b) + up_b(1, x_b) - down(1, x_b)}{2} \right\rceil$$
$$= \left\lceil \frac{i+1+\delta_1+\Phi+\delta_2+j+1+\gamma+\Phi-(i+j+1+\gamma)}{2} \right\rceil$$
$$= \left\lceil \frac{\delta_1+\delta_2+2\Phi+1}{2} \right\rceil > \Phi, \quad \text{a contradiction.}$$

If $k > i+j$, then let q_p be the rightmost unassigned exit terminal at a position $\leq q_k$. Then there are, in $P_{i,j,l}$, Φ red (resp. blue) entry terminals at positions $> q_p$ that have their assigned exit terminal to the left of q_p. We then can show by a similar argument that

$$\left\lceil \frac{up_a(q_p, x_b) + up_b(q_p, x_b) - down(q_p, x_b)}{4} \right\rceil > \Phi, \quad \text{a contradiction.} \quad \square$$

As already pointed out, Φ (and hence $\hat{d}$) can be computed in $O(n+m)$ time. This implies that we can compute an optimal assignment in time $O((n+m)\log(n+m))$ time by first finding $\hat{d}$ and then using (once) the verification algorithm.

The following theorem summarizes the results of this section.

Theorem 4.4. *The optimal density of a 2-color one-to-any problem can be computed in* $O(n+m)$ *time, and an assignment achieving the optimal density can be generated in* $O((n+m)\log(n+m))$ *time.*

5. Conclusion

Given r red and s blue entry terminals and m exit terminals, we gave an efficient algorithm for assigning an exit terminal to each entry terminal such that the resulting 2-color CRP has minimum red/blue density. The algorithm can be used in the layout design of integrated circuits.

The k-color version of this problem is worth investigating for $k \geq 3$. Generalizing Definition 4.1 to k colors is straightforward and it obviously gives a lower bound on the optimal k-color density. For the case of 3 colors, an analogous label computation in a corresponding assignment algorithm would use the dynamic version of both the 1- and 2-color problem. The difficulties arise in the proof of the lemma corresponding to Lemma 3.4. (It is currently unclear how a corresponding 'patch-up' step would have to be done.) However, we conjecture that our techniques can be extended for more than two colors.

References

[1] M.J. Atallah and S.E. Hambrusch, On terminal assignments that minimize the density, Techn. Report, CSD-TR-468, Purdue University, 1984 (submitted for publication).
[2] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms (Addison-Wesly, Reading, MA, 1974).
[3] J. Bentley, Programming Pearls, Comm. 27 (9) (1984) 865–871.
[4] Y.-K. Chen and M.-L. Liu, Three-layer channel routing, IEEE Trans. Computer-Aided Design 3 (1984) 156–163.
[5] I.S. Gopal, D. Coppersmith and C.K. Wong, Optimal wiring of movable terminals, IEEE Trans. Computers 32 (9) (1983) 845–858.
[6] R.Y. Pinter, The impact of layer assignment methods of layout algorithms for integrated circuits, Ph.D. Thesis, MIT, 1982.
[7] F.P. Preparata and W. Lipski, Three layers are enough, Proc. 23rd Annuals IEEE Foundations of Comp. Sc. Conf. (1982) 350–357.
[8] R.L. Rivest, The PI (placement and interconnect) system, Proc. 19-th Design Automation Conference (1982) 418–424.
[9] R.L. Rivest, A.E. Baratz and G. Miller, Provably good channel routing algorithms, Proc. CMU Conf. on VLSI Systems and Computations (1981) 153–159.